## **Q19** : 删除链表的倒数第 N 个结点

1. 通过 后序 递归，可以直到一个结点后边有几个结点， 若一个节点后边有n个结点，则该结点的下一个就是要删除的结点。

2. 类似于快慢指针， 让right先走n个，让后left和right一块走，这样两指针距离就是2. 那么当right走到最后一个结点,left的下一个结点就是要删除的结点。

## **Q23** : 合并K个升序链表 hard

1. 采用自底向上的分治策略， 合并整个list的结果， 等价于 先合并左侧的结果 合并右侧的结果，这两个结果的合并等于整个list合并。当一侧只有一个链表的时候就返回该链表，然后和另一侧已经合并的链表再合并。

2. 采用小顶堆的思想， 我们每次只把一个链表的第一个结点丢入堆中，那么堆顶是最小的一个结点。取出该点连接在tail上，然后把该点的下一个结点丢入堆中，这样以此类推，堆顶总能取到最小的结点。直到堆为空位置。tail后边会一个一个接上更大的元素。

3. 两种方式的时间是差不多的，小顶堆的实现会更容易一些。

## **Q24** 两两交换链表中的节点

1. 使用一个头结点来统一操作。 `while(cur != null && cur.next != null)` **cur**是第一个结点，**cur.next**是第二个节点，交换这两个结点，总是需要他们的前驱结点。

## **Q61** 旋转链表k个 : 类似于数组旋转一样。12345 -》 45123 转2个

1. 由于k可能大于链表长度，我们需要先计算长度len，此时对k求余数。然后令p先走k个， 然后q从head开始走，pq保持同步移动，当p再次走到头时，q恰好位于应该断开的位置的左侧。q.next就是新的头结点。

## **Q82** 删除排序链表中的重复元素 II

链表的题就是看起来不难，但是极容易出错的题目。

1. 一定要善用哨兵结点。

2. 断开和连接的顺序

3. 强制置空。为了避免出现循环。

## **Q138** : 复制带随机指针的链表

题意就是深拷贝： 如何进行深拷贝.

1. map : |旧结点： 新结点； 通过旧结点找到对应的新结点。

2. 第一遍，先忽略random指针，进行链表复制。 然后再遍历一遍，对random复制，通过map找到对应链表的指向。

3. 若题目变为 图的深拷贝，那么可以采用 层次遍历的特点，然后建立旧：新结点的map对应关系。一层一层的向下建立。若为当前结点建立子节点时，发现一个结点已经在map中，说明已经存在，那么不需要再建立，直接 建立 两个点的连接即可。

## **Q142** : 环形链表 II 链表分为x部分（直线部分），y部分(环部分)

1. 假设 slow走了 a个结点， fast走了b个结点 则 b = 2*a
2. 然后分析，slow和fast都走了x个结点即直线部分。 所以b-a可以减去直线部分。 b-a = ky
3. 又 b = 2*a 所以得到 a = ky. 假设k = 1, 也就是说， 直线部分，恰好等于fast,slow相遇的位置到xy交接处的长度。
4. 所以让fast在指向head, slow从相遇处出发，fast从head出发，同时都走一个结点，相遇处就是环的第一个结点。

## **Q143** : 重排链表 给定链表 1->2->3->4, 重新排列为 1->4->2->3. 给定链表 1->2->3->4->5, 重新排列为 1->5->2->4->3.

1. 将链表二分， 然后后半段逆序，然后两个链表每次各取一个，进行合并。

## 快慢指针模板

```
if(head==null || head.next==null) return head;
ListNode slow = head; //慢指针
ListNode fast = head.next; //快指针

while(fast!=null && fast.next!=null){ //快慢指针找到链表中点
    slow = slow.next; //慢指针走一步
    fast = fast.next.next; //快指针走两步
}
// slow停留在 12345停在3， 1234停在2，
```

## **Q148** : **排序链表** 要求 时间o(nlgn), 空间o(1) **链表的归并排序**

1. 可以对比几种常用的排序方法，nlgn的有快排，但是链表无法进行从右到左的访问，所以不可行。
2. 另外还有插入排序，虽然插入结点，不需要移动其他的结点，但是在查找插入位置时总是需要平均n/2的查找时间。所以总的复杂度是o(n^2)
3. o(nlgn的排序方法仅剩，归并排序，自底向上的归并排序) 使用递归的归并排序。
4. 对start-end排序。则通过快慢指针找到中点mid , 先对start-mid, mid-end排序， 然后将两个有序链表进行合并。
5. 当两个链表都只有0个或者1个结点时，就是有序链表，使用merge函数合并两个链表。

## **Q206** : 反转链表： 头插法，最好用。最不容易出错。

**Q234** : 回文链表 : 快慢指针找中点，然后逆序后半段，和前半段比较。注意 要让后半段结点数>=前半段。